International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
©2015 INPRESSCO®, All Rights Reserved
Available at http://inpressco.com/category/ijcet

Research Article

Query Subtopic Mining from Search Log Data

Namrata Gadkari†*, Sylvester Savio Raj† and Harshad Raka†

†Department of Computer Engineering, Fr. Conceicao Rodrigues College of Engineering, Bandstand, Bandra (W), Mumbai, Maharashtra, India

Accepted 20 June 2015, Available online 22 June 2015, **Vol.5, No.3 (June 2015)**

Abstract

The World Wide Web has become an integral part of our lives. This large store of data is accessed through search engines. Hence, search engines play a very important role in accessing information. But what the search engines lack is an efficient mechanism to return data which is relevant to the users. The queries fired by users tend to be ambiguous and multifaceted. Deriving the most likely facet from the queries and thus presenting relevant results to the user is a difficult task. Consider a user fires a query which is a homonym, the user has to sift through this huge list of results to find the links relevant to his search. User's past behavior is an effective parameter to determine the user's probable search intent. The user is likely to be interested in a particular facet of a multifaceted query, if he has searched for it in the past. Our method deals with returning customized results and suggestions, to users based on their previous search history, in response to a query. For this we refer to the user's search logs to obtain knowledge about his interests and based on this knowledge return customized and user relevant results.

Keywords: *Web Log Mining, Suffix Tree Clustering, Cosine Similarity, Agglomerative Clustering, Page Re-Ranking.*

1. Introduction

Most web queries are short and unclear. Some users do not choose appropriate words for a web search, and others knowingly or unknowingly omit specific terms needed to clarify search intents, because it is not always easy for users to express theirs search intents explicitly through keywords. The user queries are hence insufficient to determine the exact intent of the search. This intention gap between the user's search intent and the queries, results in queries which are ambiguous and/or broad. For ambiguous queries, users may get results which deviate from their intents whereas for broad queries, results may not be as specific as users expect.

While search engines are definitely good for certain search tasks such as searching for the homepage of an organization, they may be less effective for satisfying broad or ambiguous informational queries. The results of various subtopics of a query will be typically mixed together in the ranked list returned by the search engine, thus implying that the user may have to sift through a large number of irrelevant links.

In this report, we have proposed a method to overcome this problem. Our technique makes use of analysis of user behavior to predict the most probable results. User's past behavior is an effective parameter to determine the user's probable search intent. The search logs of a user prove to be an excellent way of acquiring knowledge about search patterns and interests of the user.

The proposed method makes use of suffix tree. Each node in the tree may have overlapping URLs. On receiving an input query from a user, the main aim is to provide the user a list of most relevant results which are ranked in an order taking into consideration the user's previous searches. Along with the results, the system also provides suggestions to the users. The whole idea is to provide a user customized search results which are more relevant, convenient and time saving to go through.

2. Literature Survey

The system proposed by Hang Li, et al. focuses mainly on clustering queries by identifying the major senses and facets. This eliminates the ambiguity and is an effective way to deal with multifaceted nature of web queries. The paper describes two interesting phenomena of user behavior: One subtopic per search and Subtopic clarification by additional keywords.

The system proposed by Oren Zamir, et al. focuses on the identification of unique requirements of document clustering of Web search engine results, the definition of STC - an incremental, $O(n)$ time clustering algorithm that satisfies these requirements, and the first experimental evaluation of clustering algorithms on Web search engine results, forming a baseline for future work. It revolves around the implementation of Suffix Tree Clustering to meet the requirements of

*Corresponding author: **Namrata Gadkari**

relevance, browsable summaries, overlap, snippet-tolerance, speed and incrementality.

The research work proposed by Neeraj Raheja, et al. describes an approach for web usage mining based upon web log partition. The research provides a system which is less time consuming and provides popular results in accordance with the existing approach.

3. Proposed system

3.1 Implementation details

3.1.1 Search Log Processing

An important key issue in query log mining is the pre-processing of logs in order to produce a good basis of data to be mined. An important step in usage analysis is thus the data preparation. This step includes: parsing the log, extracting the URLs and extracting the query.

Figure 1 shows a fragment of the AOL query log. Each row of this query log consist of records collecting five fields:
i) The ID referring to the user issuing the query
ii) The issued query
iii) The time the query was issued to the search engine
iv) The position of the clicked result in the results page
v) The host of the clicked document.

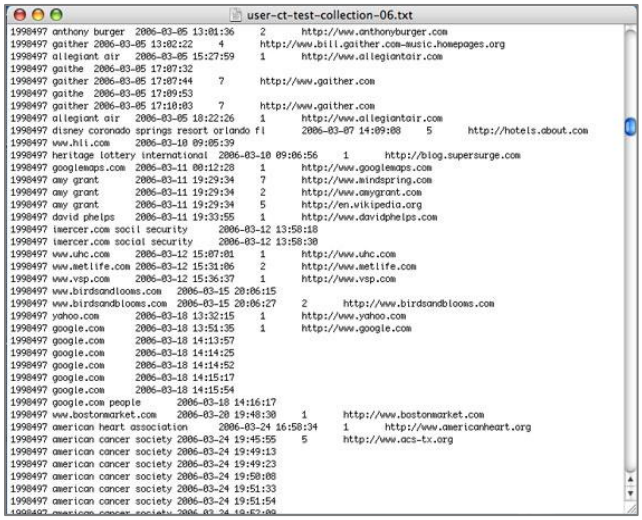

Fig.1 AOL query log

3.1.2 Suffix tree construction

The suffix tree is defined as: The tree has exactly n leaves numbered from 1 to n. Except for the root, every internal node has identifiers. In our case, each node in the suffix tree has an identifier field (keyword) which can store a set of URLs. Every internal node has children except the root node and the leaf nodes.

We construct the suffix tree using following steps:

1. The query and URLs extracted from the search log is checked to determine whether it is already present in the tree.
2. The query is split into words so that each word may form a keyword.
3. If the keyword and the URL are absent then a new node is created and an identifier and the URL is added.
4. Each keyword is made a node and the URL is assigned to each of these nodes.
5. If the keyword is present but the URL is not present then the keyword is searched in the tree and URL is added to the node identified by that keyword.
6. If both are present then no action is performed.
7. A frequency counter, which stores URLs and their frequency is used for re ranking of the results.

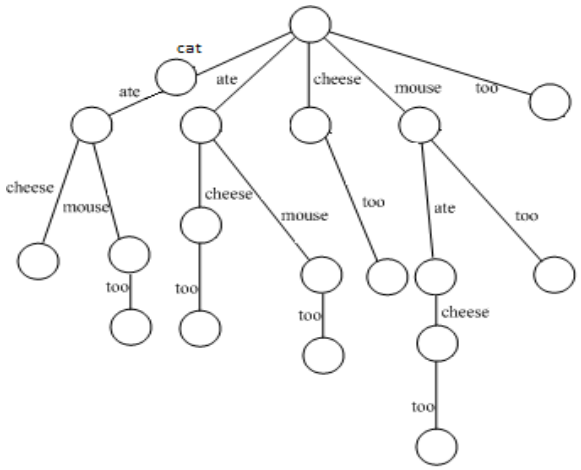

Fig.2 Example of suffix tree

3.1.3 Retrieval of URLs

The retrieval process begins with the user entering a query. The query entered by the user can be a single word or it can be a sentence.

- If the query is a single word it is searched in the suffix tree: each node's identifier is compared with this query. If it is found, the set of URLs in this node is returned as a result to the user.
- If the query is a sentence then it is split into words and each word is considered: the words are compared with each node's identifier till found. If it is found, the set URLs in this node is returned as a result to the user.
- The URLs that are returned may be of the same website or just different pages of the same website. To avoid returning such redundant results we use Cosine Similarity to identify such URLs and return only one URL that is of the main website.
- The URLs that are returned are rearranged as per their frequency as user may be more interested in these URLs more than any other URLs.

3.1.4 Giving suggestions

The URLs retrieved is given to the user but along with it helpful suggestions can be provided. To provide suggestions, we consider the node whose identifier matches the query word we considered and whose URLs are returned. We get that identifiers of the children of this particular considered node. These identifiers are good suggestions as they are related to the query word. By using suffix tree and this suggestion method we avoid formal clustering which requires additional resources and time.

Suggestions:

check

engine

clutch

diesel

petrol

gas

price

fuel

Fig.3 Example of suggestions

3.2 Algorithm

Step 1: Initialize the frequency hash map.

Step 2: Parse the search log and extract query and URL.

Step 3: Check whether query and URL are present in the tree.

Step 4: If both are not present then add the URL to the frequency hash map along with 1 and if the query is a single word then create a single node and the query becomes the identifier of the node. Whereas if the query is a sentence break it into words and for each word create a node and assign the word as the identifier for that node, also add the URL and each word that follows another word in the sentence is represented as child in the tree.

Step 5: If the query is present and URL is absent then add the URL to the frequency hash map along with 1 and if the query is a single word then add the URL to the node with the identifier same as the query whereas if the query is a sentence break it into words and for each word search the tree and add the URL to each node that is found.

Step 6: If both are present then the URLs frequency value is updated in frequency counter.

Step 7: Enter a query, if the query is a single word it is searched in the suffix tree i.e. Each node's identifier is compared with this query if it is found the set URLs in this node is returned as a result to the user whereas if the query is a sentence then it is split into words and each word is considered, each word is compared with each node's identifier till found and if it is found the set URLs in this node is returned as a result to the user.

Step 8: The URLs that are returned may be same or different pages of the same website to avoid such results we use Cosine Similarity to identify such URLs and return only one URL, in cosine similarity the URLs are split into words and depending on the frequency of each word and the formula for cosine similarity, it returns a value based on the overall similarity of the URLs, if it is above the threshold value then only one URL is given to the user else both are given.

Step 9: After receiving the URLs re ranking is performed based on the values of the frequency counter.

Step 10: Final results is given to user along with suggestions, these suggestions are basically the children of the node that has the identifier same as the query word we considered to get the result.

3.2 flowcharts for proposed system

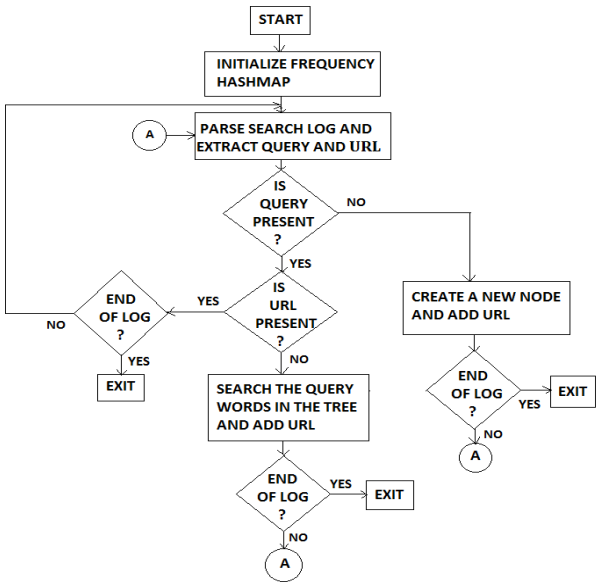

Fig.4 Search log processing and construction of suffix tree

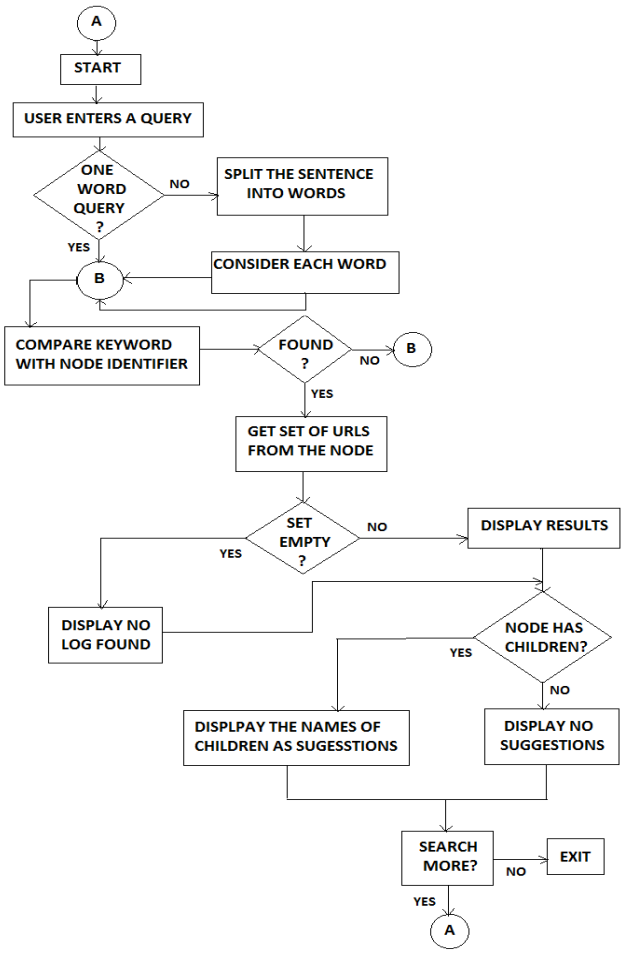

Fig.5 Retrieval and suggestions

4. Results and Discussion

The user enters the input query in the textbox alongside "SEARCH"

Fig.6 Home page

The result obtained is a set of re-ranked URLs displayed with their frequency of occurrence. It also provides suggestions.

Fig.7 Result page

The user can go back to the homepage to input a new query.

Fig.8 Link back to home page at the bottom of the results page

Query used to generate search log file:
Honda accord fuel additives check engine light.
Honda civic engine.

The proposed system allocates a node to all the words in the query as the children of the root node i.e. in Breadth First manner and thereon adds words in Depth-First manner. As the nodes are being made and put in their respective positions, all the associated URLs are also put along with them.

Consider a query

Honda
This query will be mapped with all the words in the tree by parsing it from the leftmost node at level one to the rightmost. For the query considered, the match is found at the first node itself. Hence no further search is made and the associated URLs are provided to the user. Since Accord and Civic are the children of node Honda, they are presented as suggestions to the user.

Consider another query

Honda civic

This query, since is formed of more than one word, is parsed from right to left as stated in the proposed systems algorithm. Therefore the word 'CIVIC' is used to match with the nodes at the level one. A match is found at level one. Since engine is the child of Civic, it given as suggestion and all the URLs associated with civic are displayed for the user. The advantage of parsing the query from right to left is that we can get a match while parsing the tree in Breadth First Search and not Depth First Search. Since we are limiting on the number of nodes added at level one, the chances of a hit increases for a query.

Conclusions

The project proposes an approach for clustering the user information based on the search log data saved in the log file. Using this log file as the foundation, a tree is generated based on the queries and associated URLs. Upon firing a query, the tree is parsed in Breadth First Search at level One and in Depth First Search for the further levels thereon. Based on the user's navigation pattern and queries fired, the URLs in the retrieved result set are re-ranked using frequency of visit as the base parameter. Also, additional suggestions based on the query are provided to the user. We have tested the effectiveness of this approach.

There are several other related aspects we wish to explore to upgrade our current work status. First, we have worked with only one clustering algorithm. Other similar techniques can be used to improve the efficiency of the system. Second, one level down the current working context in the tree are displayed as suggestions. Advanced and optimized suggestion generation techniques can be implemented. Most importantly, large search log data is of utmost importance for the system to work. Therefore, search log files are the pre-requisite which, at times, turns out to be a drawback.

Acknowledgment

With deep sense of gratitude we would like to acknowledge the contribution towards this research and inspiring guidance of Prof. Kalpana Deorukhkar, assistance professor, Department of Computer Engineering, Fr. Conceicao Rodrigues College of Engineering.

References

Yunhua Hu, Yanan Qian, Hang Li, Daxin Jiang, Jian Peiz, and Qinghua Zheng (2012), Mining Query Subtopics from Search Log Data, IGIR '12 Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, pp 305-314

J. Allan and H. Raghavan (2002), Using part-of-speech patterns to reduce query ambiguity, Proceedings of SIGIR'02, pp 307-31

Neeraj raheja1 and V.K.katiyar (2014), Efficient web data extraction using clustering approach in web usage mining, International Journal of Computer Science Issues, Vol. 11 issue 1 No 2, pp 216-225

Ida Mele (2013), Web Usage Mining for Enhancing Search Result Delivery and Helping Users to Find Interesting Web Content, ACM, WSDM'13, pp 765 769

Suneetha K. R., and Krishnamoorthi R. (2009), Identifying User Behavior by Analysizing Web Server Access Log File, IJCSNS International Journal of Computer Science and Network Security, Vol 9, No.4, pp 327-332

Xiao Li, Ye-Yi Wang, Alex Acero (2008), Learning query intent from regularized click graphs, SIGIR '08 Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pp 339-346

A. Z. Broder (2002). A taxonomy of web search. *Sigir Forum*,36, pp 3–10

J. Pei, J. Han, B. Mortazavi-Asl (2000), and H. Zhu, Mining access patterns efficiently from web logs, PADKK '00 Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications, pp 396-407.

L.K. Joshila Grace, V.Maheswari and Dhinaharan Nagamalai(2011), Analysis of web logs and web user in web mining , International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.1.,pp 99-110

J.Srivatsava, R.Cooley, M.Deshpande and P.N. Tan (2000), Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data, ACM SIGKDD Explorat. Newsletter, pp 12-23.

E. Ukkonen (1995). On-line construction of suffix trees. Algorithmica, 14, pp 249-60